SKYLINE-BASED TERRAIN MATCHING USING A MAX-MIN PRINCIPLE

NAGW-1333

by

C.N. Shen & Lance Page

Rensselaer Polytechnic Institute
Electrical, Computer, and Systems Engineering
Troy, New York 12180-3590

October, 1990

CIRSSE REPORT #72

Skyline-Based Terrain Matching Using a Max-Min Principle*

C.N. Shen Lance Page

Electrical, Computer, and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180

Abstract

A new approach to skyline-based terrain matching is described. It is based on selecting the longest of the minimum-distance vectors between corresponding curves from two sets of data, and is thus called "max-min matching." In the terrain matching case, one set of curves is a set of skylines generated by an autonomous vehicle by its 3-D visual sensors, and the other is generated from a terrain map database. The algorithm is being developed to perform skyline-based terrain matching for autonomous vehicle navigation in an unstructured environment such as the surface of Mars, but the approach is suitable for other types of scenes as well. Initial experiments are described, which gave promising results for the matching technique.

1 Introduction

The task of *terrain matching* is to check or refine an autonomous vehicles's estimate of its own *vantage*, that is, position and orientation of its visual sensors with respect to a "global map" of the terrain [1, 2, 3, 4]. The terrain matching task in various contexts has also been referred to as robot localization [5] and position estimation [6, 7].

Terrain matching is one of the basic subtasks of autonomous vehicle navigation. Once the vantage is established, the vehicle may perform its path selection based on its current position, the information from both the global map and its sensors, and whatever instructions have been issued to the vehicle. The autonomous navigation process for a roughly mapped, unstructured environment is further described in [2, 3].

Prior to terrain matching, the vehicle must make sensory measurements of its environment. In our case, it uses stereo vision or laser ranging to develop a local, 3-D map of the terrain, and extracts the skylines, or occluding contours from those measurements. The skylines from sensory measurements, called "local skylines," are space curves which can be expressed in parametric form with azimuth as the independent variable, and horizontal distance and vertical height as functions of the azimuth.

Similar curves called "global skylines" are generated from the global map based on an *a priori* estimate of the vantage[1]. Skyline based terrain matching uses the local and global skylines as features that are common to both the local and global maps. It compares the two sets of skylines in order to relate the local map to the global map and thus determine the vantage.

The comparison between a local skyline and global skyline consists of two main steps:

*Research performed for the Center of Intelligent Robotic Sytems for Space Exploration at Rensselaer Polytechnic Institute under NASA contract #NAGW-1333.

1. Find the longest of the minimum-distance vectors between the curves. This is called the max-min.
2. Using the max-min to determine the point-by-point correspondence between the curves, compute the average vector difference between corresponding points. This average estimates the error in the previous vantage estimate.

The vantage estimate is then updated by the estimate of its error.

Section 2 of this report describes the terrain matching problem in more specific terms. Section 3 describes the fundamental ideas behind the new approach. Section 4 describes the test curves used in our initial experiments, and the cubic splines used to represent them. Sections 5 and 6 describe the max-min and averaging computations, respectively. Section 7 concludes the report.

2 Statement of the problem

The vantage consists of six parameters: three for translation (x_v, y_v, and z_v), and three for orientation (yaw, pitch, and roll). The translational vantage parameters represent the true location of the sensors, with respect to a global coordinate system, (x^g, y^g, z^g) fixed to the terrain. A local coordinate system, (x^l, y^l, z^l), is fixed to the vehicle at its (true) vantage point. In this work, the orientation parameters of the vantage are assumed known; for simplicity, we assume that the vehicle's yaw, pitch, and roll are zero, so that the local and global coordinate systems are related by a mere translation.

The goal of the algorithm is to compute accurate $\hat{x_v}$, $\hat{y_v}$, and $\hat{z_v}$, estimates of the unknown vantage parameters. Since the algorithm is iterative, $\hat{x_v}^i$, $\hat{y_v}^i$, and $\hat{z_v}^i$ are the vantage estimates after the ith iteration. $\hat{x_v}^0$, $\hat{y_v}^0$, and $\hat{z_v}^0$ are based on a previous position estimate of the vehicle, and on the dead reckoning path which it has approximately followed from that position.

2.1 Problem description in detail

A coordinate frame called the "star" ($\star$) system originates at $(\hat{x_v}^i, \hat{y_v}^i, \hat{z_v}^i)$ in global coordinates; that is, at the estimated vantage point. (See Figure 1.) It is emphasized that the origin of the star system, $(\hat{x_v}^i, \hat{y_v}^i, \hat{z_v}^i)$ is known exactly, but that of the local system, (x_v, y_v, z_v) is not. Denote the error in the vantage estimate after i iterations as (x_0^i, y_0^i, z_0^i):

$$\begin{aligned} x_v &= \hat{x_v}^i + x_0^i \\ y_v &= \hat{y_v}^i + y_0^i \\ z_v &= \hat{z_v}^i + z_0^i \end{aligned} \tag{1}$$

The task of the algorithm in the ith iteration is to compute $\hat{x_0}^{i-1}$, $\hat{y_0}^{i-1}$, and $\hat{z_0}^{i-1}$, the estimates of the previous error. From these values, the vantage estimate is updated as:

$$\begin{aligned} \hat{x_v}^i &= \hat{x_v}^{i-1} + \hat{x_0}^{i-1} \\ \hat{y_v}^i &= \hat{y_v}^{i-1} + \hat{y_0}^{i-1} \\ \hat{z_v}^i &= \hat{z_v}^{i-1} + \hat{z_0}^{i-1} \end{aligned} \tag{2}$$

The iteration continues until sufficient accuracy has been obtained in the vantage estimate.

We now turn to the question of computing $\hat{x_0}^i$, $\hat{y_0}^i$, and $\hat{z_0}^i$. From here forward, the i superscripts will be dropped; it may be assumed that we are in the $i+1$th iteration, calculating values with the i superscript. For instance, we are given $\hat{x_v}^0$, $\hat{y_v}^0$, and $\hat{z_v}^0$, and in the first iteration are computing $\hat{x_0}^0$, $\hat{y_0}^0$, and $\hat{z_0}^0$.

To compute the vantage error estimates, skyline curve data from two sources are used:

1. A set of "local skylines" extracted from the visual sensors on the vehicle. These curve data are expressed in cylindrical coordinates about the local coordinate system.

2. A set of "global skylines" which are expected to be visible from the estimated vantage, are generated from a rough, global map of the terrain. The global skylines are expressed in cylindrical coordinates about the star coordinate system. A method of generating global skylines from a global map is described in [1].

The global skylines are taken as estimations of the local skylines in 3-D space, in spite of two differences:

1. Errors in the global map cause the actual (local) and predicted (global) skylines to differ.

2. Different vantages cause different sets of skylines to result, even if the sources of information are identical, and so an error in the vantage would cause the global skylines to differ from the local even if the map was perfect.

These differences are ignored. If the vantage estimate improves in each iteration of the algorithm, then the differences resulting from the second cause listed will diminish.

In order to perform computations with curves which come from two different coordinate systems, the transformation between which (i.e. the translation of x_0, y_0, and z_0) is not known, a third coordinate system is introduced: a "dagger" coordinate system which is coincident with the star coordinate system. The curves expressed in dagger coordinates are translated copies of the local skylines; the relationship of the dagger curves to the dagger system, shown in Figure 2, is identical to that of the local curves to the local coordinate frame.

An additional coordinate system will be used, in graphically evaluating the resulting $\hat{x_0}$, $\hat{y_0}$, and $\hat{z_0}$ from the algorithm: the double-dagger (‡) system is yet another translated version of the local system and local skylines. This system originates at the global coordinates $(\hat{x_v} + \hat{x_0}, \hat{y_v} + \hat{y_0}, \hat{z_v} + \hat{z_0})$, shown in Figure 3, which could also be called: the vantage estimate for the next iteration; the "star" origin for the next iteration; or the *post priori* vantage estimate for this iteration. Plots of the star system and the double-dagger system together demonstrate how well the algorithm "matched" the local and global skylines.

2.2 Scope of experiments

The experiments described in this report involved a single pair of corresponding curves, instead of two sets of curves representing two versions of an entire scene. This eliminated the major step of evaluating the *curve correspondence* [1] between the two sets of curves.

In addition, these curves were not generated from a global map or from visual data; for testing purposes, they were simply derived from an analytical function described in Section 4.1. The two main differences between the curves, listed above, thus do not exist for the test case. The portions of the curves which did correspond, matched almost exactly in the result- the error that did appear arose from the spline interpolation, and was only slight.

The test began with data for the star curve, representing the global skyline for an "incorrect" vantage estimate; and for the dagger curve, representing a local skyline already translated from its unknown but true position, to the known but incorrect vantage estimate. It proceeded to calculate $\hat{x_0}$, $\hat{y_0}$, and $\hat{z_0}$ so that the local skyline in the double-dagger position (retranslated by the inverse error estimate) matched the star curve.

Only one "iteration" of the technique is performed. (The method performs sufficiently well under the test conditions that only one iteration is necessary for an accurate result.)

2.3 Definitions

Several concepts will be used for relating points on one curve to points on the other throughout this report. The first and and most important is that of a *corresponding point.* Two points on different curves are said to correspond if they represent the same physical point, in this case on a terrain.

A *discrepancy* is defined as the shortest vector from a specified point on one curve, to the other curve.

A *normal-to-intersect vector* is a path along the normal line through some point on one curve, to the normal line's intersection with the other curve. For normal curves from some points, this intersection does not exist. The distance along the normal line, in the direction which immediately increases the radius of the line from the origin, is called ϵ; a normal-to-intersect from an "inner curve" (smaller $\rho(\theta)$) to an "outer curve" (larger $\rho(\theta)$) has a positive ϵ, and a normal-to-intersect from an "outer curve" to an "inner curve" has a negative ϵ.

A discrepancy vector or normal-to-intersect vector goes from one curve to another. The point on the curve, where the vector is computed *from* shall be called its base point, and the point on the other curve which the vector points *to* shall be called its terminating point. The curves may also be called the base curve and terminating curve, respectively.

Throughout this report, each cylindrical coordinate system is related to its corresponding rectangular coordinate system with θ measured counter-clockwise from the x-axis; ρ as the length of the horizontal projection; and z the same as the rectangular z. Mathematically,

$$\begin{aligned} \theta &= \operatorname{atan2}(y.x) \\ \rho &= \sqrt{x^2+y^2} \\ z &= z \end{aligned} \tag{3}$$

where the atan2(y,x) function is a special version of $\arctan(y/x)$, which insures that the resulting angle is in the appropriate quadrant.

3 The max-min principle

We begin our development with the max-min principle[1], which applies to the special case of two curves which are completely identical except for a translation between them. In this special case, every point on one curve has a corresponding point on the other. Later we will relax this restriction, to allow for a pair of curves whose end points might not correspond, and will have to consider possible exceptions to the original principle. The max-min principle is as follows:

Max-min Principle: *The minimum distance from any point on a curve, to a translated version of the same curve, is less than or equal to the length of the translation.*

This is justified as follows: there is at least one point— the corresponding point— on the translated curve which will be the length of the translation away from a point on the original curve. There may also be other points on the translated curve which are closer than the corresponding point. Therefore, the minimum distance will be less than or equal to the length of the actual translation.

We postulate that the longest of these minimum-distance vectors, called the max-min, will come close to the actual translation between the curves. (There are certain conditions under which the discrepancy is guaranteed to equal the translation or the negative of the translation, although we have not yet enumerated them.) The max-min could be used an estimate of the translation itself, but in fact only its direction is

[1]The term "max-min" in this context has nothing to do with max-min game theory in the field of artificial intelligence.

used, and the component of translation in that direction is calculated with an averaging scheme described in Sect. 6.

Calculating the discrepancies means finding the minimum (Euclidean) distance between some point (on one curve) to another curve. With piece-wise polar cubic curves (i.e. cubic splines in polar coordinates) it is difficult to compute the discrepancies. Numerically it is expensive, because multiple minima and maxima of distance from the point may exist along the curve, and in polar space the whole curve may have to be searched. However, it is rather straightforward to numerically compute normal-to-intersect distances, ϵ, from one curve to another curve, and these distances are in most cases good substitutes for the discrepancies. Whereas a discrepancy vector should be normal to its terminating curve (marking a local minimum in distance), a normal-to-intersect vector is normal to its base curve. By finding the point where a normal line of one curve intersects the other curve, we in most cases obtain a discrepancy vector in the *reverse direction.* (In such cases, the normal-to-intersect vector marks the shortest path from its terminating point to its base curve.) For normal-to-intersects, the search for each ϵ is along a line, and is thus numerically quite straightforward.

Insofar as the normal-to-intersect measurements substitute for discrepancy measurements, the principle above still applies. However, to maximize our chances that at least one ϵ is approximately as long as the actual translation, we make normal-to-intersect measurements in both directions— from the original curve to the translated curve, and vice-versa. In short, we propose to use the maximum normal-to-intersect measurement, to estimate the direction of the translation.

3.1 Max-min exceptions

This section highlights two kinds of exceptions to the max-min principle: exceptions that are possible due to non-corresponding endpoints, and exceptions possible due to using normal-to-intersect measurements instead of discrepancy measurements.

When matching skylines, we must be prepared for the endpoints of the curves to not correspond to each other. That is, while the curves must share some common, corresponding portion of a skyline, one curve may be "longer" than the other on either or both ends. (If the endpoints were known to correspond to each other, then the translation could be easily calculated from endpoints only.) In fact, the endpoints are the most sensitive parts of the skylines (often located at visual inflection points), and will most often *not* correspond to eachother due to data noise and vantage error.

If discrepancies themselves were being measured (as opposed to normal-to-intersects), then discrepancies near noncorresponding endpoints could be longer than the translation between the curves, and have little to do with the direction or translation between the curves; this could disrupt the entire algorithm. A way to avoid this case would be to discard any discrepancies whose terminating point is an endpoint of its terminating curve. A nearly equivalent condition would require the discrepancy vector to be normal to the terminating curve at its terminating point, but merely comparing it to the endpoints is easier.

Even following this rule still allows for cases, for example see Figure 4, where noncorresponding endpoints can cause extra-long discrepancies. Such cases, however, are expected to be extremely rare, and have not been further addressed.

A different type of exception is made possible by using normal-to-intersect measurements in place of discrepancy measurements. This exception, illustrated in Figure 5, may result from a noisy base curve, or from a pair of curves with a distinct irregularity.

The interpolation scheme is typically smooth (C-2 continuous), and filtering techniques may be applied in the future as necessary, in order to prevent noise from causing this type of exception. The case of a distinct irregularity— referring in the right-hand side of Figure 5 to the "bump" common to both curves— may demand more attention in the future. To a human matching the curves intuitively, an irregularity common to both curves provides valuable clues; an autonomous matching scheme should be able to at least tolerate, if not thrive upon, such potentially valuable information. (The discrepancy measurements do in fact take advantage of such an irregularity, because the peak in discrepancy length, taken in a certain direction, is very sharp.)

One way to handle this exception may be to make sure that the distance from a normal-to-intersect's terminating point to its base curve is in fact a local minimum at its base point compared to its neighbors on the base curve, and not a local maximum. Merely the fact that it is normal to its base curve allows for either a maximum or a minimum, but to take the place of a "reverse discrepancy" it must be a minimum. A test for this compares the curvature of the base curve at the base point, to the measured ϵ, but this test was not implemented for the experiments described in this paper.

4 Interpolation with natural cubic splines

The algorithm receives as input two curves— a dagger curve and a star curve— and each is represented as a sequence of points in cylindrical coordinates. In our experimental case, the sequences of points describing the curves are generated from analytic functions described below; in the future, the curve data will come from a global map-based skyline generator described in [1], or even real visual data. Regardless of the source, the algorithm's first step is to interpolate each sequence of points into a smooth curve, using natural cubic splines. The splines parameterize ρ and z separately in terms of θ, since θ is the natural abscissa for visual data on a $2\frac{1}{2}$-D unstructured terrain[2].

Once the splines were calculated, the curves were available as continuous functions, $\rho^\dagger(\theta^\dagger)$, $z^\dagger(\theta^\dagger)$ for the dagger curve, and $\rho^\star(\theta^\star)$, $z^\star(\theta^\star)$ for the star curve. The interpolated functions were only defined between the endpoints of the curves, and the discrete samples from which the splines were interpolated became transparent to the algorithm.

4.1 Experimental input

In our initial experiments we used the following analytical functions for the dagger curve:

$$\begin{aligned} \rho_{sp}(\theta) &= \rho_c e^{a\theta} \\ z_{sp}(\theta) &= z_c \cos\left[1.5(\theta - 45^\circ)\right] \end{aligned}$$

where ρ_c, a, and z_c are constants. The analytic functions were sampled at even increments of $\theta^\dagger$ over a certain range, using $\rho^\dagger(\theta^\dagger) = \rho_{sp}(\theta^\dagger)$ and $z^\dagger(\theta^\dagger) = z_{sp}(\theta^\dagger)$; these discrete points were fed into the cubic spline generator described in the next section.

The experimental star curve consisted of the same analytical function, sampled across a different range of θ than the dagger curve, then translated by an x_0, y_0, and z_0. A different range was used, so that the endpoints of the two curves did *not* correspond to each other; the translation parameters were the values which the algorithm was designed to estimate. The star points were generated for discrete values of θ_i as follows:

$$\begin{aligned} \theta_i^\star &= \arctan\left(\frac{\rho_{sp}(\theta_i)\sin\theta_i + y_0}{\rho_{sp}(\theta_i)\cos\theta_i + x_0}\right) \\ \rho_i^\star &= \sqrt{(\rho_{sp}(\theta_i)\sin\theta_i + y_0)^2 + (\rho_{sp}(\theta_i)\cos\theta_i + x_0)^2} \\ z_i^\star &= z_{sp}(\theta_i) + z_0 \end{aligned}$$

[2]A $2\frac{1}{2}$-D unstructured terrain refers to a terrain on which the ground height is an arbitrary, continuous function of horizontal position. Caves or overhangs violate such a terrain model.

The star curve's discrete samples were likewise interpolated with cubic splines.

Figure 6 shows the experimental star and dagger curves, with the discrete points from which they were generated highlighted, and the spline interpolation described in the next section already performed. For these curves, $\rho_0 = 10.0$, $a = 0.4$, and $z_c = 3.$; $x_0 = 5.0$, $y_0 = 2.5$, $z_0 = 2.5$. The dagger curve had a $\theta^{\dagger}$ range of 10° to 100°. Before translating, the range of θ used for generating the star curve was from −25° to 90°, and the resulting range of θ^{*} was from about 1.1° to 76.8°. Each curve was sampled with ten points prior to cubic spline interpolation.

4.2 Applying `spline()` and `splint()`

The cubic spline interpolation is performed with the `spline()` and `splint()` subroutines of [8]. Interpolation of $\rho(\theta)$ is described here; the procedure for interpolating $z(\theta)$ is identical.

The `spline()` routine calculates a list of second derivatives, $\rho'' = d^2\rho/d\theta^2$, to accompany the discrete θ and ρ values. The spline can then interpolated (with function `splint()`) as:

$$\rho(\theta) = A\rho_j + B\rho_{j+1} + C\rho_j'' + D\rho_{j+1}'' \tag{4}$$

where θ_j, ρ_j, ρ_j'', θ_{j+1}, ρ_{j+1}, and ρ_{j+1}'' are the stored spline parameters at each end of the interval containing θ, and

$$\begin{aligned} A &= \frac{\theta_{j+1} - \theta}{\theta_{j+1} - \theta_j} \\ B &= \frac{\theta - \theta_j}{\theta_{j+1} - \theta_j} \\ C &= \frac{1}{6}(A^3 - A)(\theta_{j+1} - \theta_j)^2 \\ D &= \frac{1}{6}(B^3 - B)(\theta_{j+1} - \theta_j)^2 \end{aligned}$$

This interpolation has the following properties:

1. Each segment is a polar cubic polynomial, i.e. a cubic polynomial in polar coordinates.
2. The spline passes through all points in the list of discrete samples.
3. The spline is C2-continuous, meaning it is continuous, and has continuous first and second derivatives.
4. The second derivative, ρ'' is constrained to be zero at the two endpoints of the interpolated curve. This property makes it a so-called "natural spline."

The first derivative may be calculated as:

$$\frac{d\rho(\theta)}{d\theta} = \frac{\rho_{j+1} - \rho_j}{\theta_{j+1} - \theta_j} - \frac{3A^2 - 1}{6}(\theta_{j+1} - \theta_j)\rho_j'' + \frac{3B^2 - 1}{6}(\theta_{j+1} - \theta_j)\rho_{j+1}'' \tag{5}$$

5 Computing the max-min

Computing the max-min means selecting the longest normal-to-intersect in either direction, from the dagger curve to the star curve or vice-versa.

5.1 Computing normal-to-intersects

We describe the normal-to-intersect procedure, for intersecting the normal to a point on the dagger curve with the star curve. Computing a normal-to-intersect in the reverse direction is identical except for swapping the curves.

The first step is to find an expression for the line which is normal to the dagger curve at some $\theta = \theta_n$ (n subscript is for "normal point"). Define ϕ as the angle that the dagger curve at $(\theta_n, \rho^\dagger(\theta_n))$ makes with a curve of constant ρ through that point.

$$\phi = \arctan\left(\frac{d\rho^\dagger(\theta_n)}{\rho^\dagger(\theta_n)d\theta}\right)$$

The normal line will point in the direction of $(\theta_n - \phi)$.

Next, parameterize this line in terms of ϵ, which is the distance along the line in the direction of increasing ρ; the direction along the line of (initially) decreasing ρ is the $-\epsilon$ direction. Letting the l subscript denote "line," the x and y coordinates of the normal line will be:

$$\begin{aligned} x_l(\epsilon) &= x_n + \epsilon c_x \\ y_l(\epsilon) &= y_n + \epsilon c_y \end{aligned}$$

where

$$\begin{aligned} x_n &= \rho^\dagger(\theta_n)\cos\theta_n \\ y_n &= \rho^\dagger(\theta_n)\sin\theta_n \end{aligned}$$

and

$$\begin{aligned} c_x &= \cos(\theta_n - \phi) \\ c_y &= \sin(\theta_n - \phi) \end{aligned} \tag{6}$$

Next, parameterize ρ and θ on this line:

$$\rho_l(\epsilon) = \sqrt{(x_n + c_x\epsilon)^2 + (y_n + c_y\epsilon)^2} \tag{7}$$

$$\theta_l(\epsilon) = \text{atan2}((x_n + c_x\epsilon), (y_n + c_y\epsilon)) \tag{8}$$

At the point where the normal line intersects the star curve, $\rho_l(\epsilon) = \rho^\star(\theta_l(\epsilon))$. Thus, define

$$f(\epsilon) = \rho^\star(\theta_l(\epsilon)) - \rho_l(\epsilon)$$

our task is to search for $\epsilon : f(\epsilon) = 0$. This search is performed by applying the **zbrac()** and **zbrent()** subroutines of [8].

5.2 Applying zbrac() and zbrent()

The **zbrac()** routine brackets a root, meaning it attempts to find an ϵ_1 and ϵ_2 such that $f(\epsilon_1)$ and $f(\epsilon_2)$ have opposite signs. Under this condition, a root is guaranteed to exist between ϵ_1 and ϵ_2. It is initialized with the interval from 0 to $f(0) = \rho^\star(\theta_n) - \rho^\dagger(\theta_n)$, and expands if necessary until it brackets a root. If it fails to bracket a root after a specified number of expansions, then the normal-to-intersect is considered not to exist for this θ_n. The **zbrent()** routine contracts the interval ϵ_1, ϵ_2 surrounding the root until it is sufficiently small to provide the required precision. Additional details of `zbrac()` and `zbrent()` are well documented in [8].

The subroutine which calculates $f(\epsilon)$ uses a value for $\rho^\star(\theta_l(\epsilon))$ even when $\theta_l(\epsilon)$ is outside the star curve's valid range, by extrapolating the cubic equation of the nearer end section of the star curve to the invalid $\theta_l(\epsilon)$. The algorithm checks that the solution makes a valid intersection with the star curve, *after* `zbrent()` returns a root; if the resulting $\theta_l(\epsilon)$ lies outside the valid bounds of the star curve, then the normal-to-intersect is considered not to exist for this θ_n, just as if a solution had not been bracketed.

5.3 Selecting the maximum ϵ

The normal-to-intersects are computed from a number of locations along the dagger curve, at equal increments of $\theta^\dagger$, and from a number of locations on the star curve, at equal increments of $\theta^\star$. The normal-to-intersects for our test curves are shown in Figures 7. For either direction, graphing the normal-to-intersect distances produces an *epsilon profile*. Epsilon profiles for our pair of test curves are shown in Figure 8.

The normal-to-intersect vector with the overall biggest magnitude (absolute value) is selected as the max-min; it is the longest normal-to-intersect vector. The max-min is represented by its ϵ, and its c_x and c_y of Equ. 6 which represent the direction of the normal-to-intersect vector. The direction, in terms of whether it is a dagger-to-star normal-to-intersect or a star-to-dagger curve normal-to-intersect, is also noted.

6 Estimating the translation from the max-min

This section describes how the translation (x_0, y_0, z_0) can be estimated, once the max-min is computed.

6.1 Horizontal translation directly from the max-min

A rough but simple way to estimate x_0 and y_0 from the max-min is to use the longest normal-to-intersect vector directly as the translation vector. If it was a dagger-to-star normal-to-intersect, then

$$\begin{aligned} \hat{x_0} &= \epsilon c_x \\ \hat{y_0} &= \epsilon c_y \end{aligned}$$

If the max-min vector was a star-to-dagger normal-to-intersect, then

$$\begin{aligned} \hat{x_0} &= -\epsilon c_x \\ \hat{y_0} &= -\epsilon c_y \end{aligned}$$

In our experimental case the max-min was a star-to-dagger normal-to-intersect, starting at $\theta^\star = 48.6°$, and having $\epsilon = -5.590$. The horizontal and vertical components of this vector, when reversed to account for the star-to-dagger direction, are shown in Table 1; Figure 9 shows the resulting match.

6.2 Horizontal translation calculated with averaging

A better estimate of the translation is possible by retaining the direction of the max-min but not its length; the length of the translation is then estimated as the *average* of the distance between the curves, taken in this direction.

	x_0	y_0	z_0
Actual translation	5.0	2.5	2.5
Direct estimation (Sect. 6.1)	5.0215	2.4564	—
Averaged estimation (Sect. 6.2 & 6.3)	4.9898	2.5170	2.4998

Table 1: Summary of estimation results.

In this case,

$$\hat{x_0} = c_x \frac{1}{\left(N_{\dagger\star} + N_{\star\dagger}\right)} \left(\sum_{i=1}^{N_{\dagger\star}} \Delta_{\dagger\star i} - \sum_{i=1}^{N_{\star\dagger}} \Delta_{\star\dagger i} \right) \tag{9}$$

$$\hat{y_0} = c_y \frac{1}{\left(N_{\dagger\star} + N_{\star\dagger}\right)} \left(\sum_{i=1}^{N_{\dagger\star}} \Delta_{\dagger\star i} - \sum_{i=1}^{N_{\star\dagger}} \Delta_{\star\dagger i} \right) \tag{10}$$

where each Δ_i is a *direction-to-intersect* measurement from one curve to another; $N_{\dagger\star}$ and $N_{\star\dagger}$ are the number of direction-to-intersects which intersected in the respective directions. Again, note that the star-to-dagger measurements are negated for purposes of calculating $\hat{x_0}$ and $\hat{y_0}$. Computationally, the direction-to-intersects are equivalent to normal-to-intersects, but their directions represented by c_x and c_y are pre-specified rather than calculated with Equ. 6 for each measurement.

Direction-to-intersects are calculated from the dagger curve at equal increments of $\theta^\dagger$, and from the star curve at the same number of increments. However, only the direction-to-intersects that do intersect the other curves are included in $N_{\dagger\star}$ and $N_{\star\dagger}$, and are used in Eqs. 9 and 10. The averaging acts to filter out noise present in the max-min calculation.

The results of applying Eqs. 9 and 10 are shown in Table 1 and Figure 10. The advantages of averaging the direction-to-intersects over using the max-min directly are expected to be more dramatic in cases with more noise than the example in this paper.

6.3 Estimating the vertical translation

This section examines the evaluation of $\hat{z_0}$.

The direction of the max-min effectively provides a rule for point correspondence between the two curves; Eqs. 9 and 10 in effect measure the x and y components of the average horizontal distance between corresponding points according to this rule. Once the direction-to-intersects are computed, the vertical distances are also readily available; the z values at the base and terminating points of each direction-to-intersect vector can be computed from the $\theta^\dagger$ and $\theta^\star$ values at these points. $\hat{z_0}$ is computed as the average difference between these heights:

$$\hat{z_0} = \frac{1}{N_\Delta} \sum_{i=1}^{N_\Delta} \left(z^\star(\theta^\star_{\Delta i}) - z^\dagger(\theta^\dagger_{\Delta i}) \right) \tag{11}$$

In this equation, N_Δ equals $N_{\dagger\star} + N_{\star\dagger}$ of Eqs. 9 and 10; the summation is over all connected direction-to-intersects.

The $\theta^\dagger_\Delta$ and $\theta^\star_\Delta$ denote the θ values at the base and terminating points of a direction-to-intersect vector (not necessarily respectively). These values are readily available during the direction-to-intersect computations for Eqs. 9 and 10. Consider for instance a dagger-to-star direction-to-intersect: $\theta^{\dagger}_{\Delta}$ is pre-specified, and $\theta^{\star}_{\Delta}$ is given (from Equ. 8) as:

$$\theta^{\star}_{\Delta} = \theta_l(\Delta) = \text{atan2}\left((x_n + c_x\Delta), (y_n + c_y\Delta)\right)$$

where

$$\begin{aligned} x_n &= \rho^{\dagger}(\theta^{\dagger}_{\Delta})\cos\theta^{\dagger}_{\Delta} \\ y_n &= \rho^{\dagger}(\theta^{\dagger}_{\Delta})\sin\theta^{\dagger}_{\Delta} \end{aligned}$$

c_x and c_y represent the direction of the direction-to-intersects; and Δ is the direction-to-intersect distance. Eqs. 9, 10, and 11 are thus calculated simultaneously with each other, one direction-to-intersect at a time.

The results of matching the vertical data are shown in Table 1 and Figure 11. This figure deserves some explanation: the star curve is still plotted as $z^{\star}(\theta^{\star})$ vs. $\theta^{\star}$, as in Figure 6. The double-dagger z function, however, is also plotted as a function of $\theta^{\star}$, and this requires a transformation relating the two systems, specifically $\theta^{\ddagger}$ to $\theta^{\star}$. We call this the correspondence relationship, denoted with a c subscript, and the horizontal geometry of Figure 10 indicates that:

$$\theta^{\ddagger}_{c}(\theta^{\star}) = \text{atan2}\left((\rho^{\star}(\theta^{\star})\sin\theta^{\star} - \hat{y}_0), (\rho^{\star}(\theta^{\star})\cos\theta^{\star} - \hat{x}_0)\right) \tag{12}$$

The double-dagger curve in Figure 11 is thus $z^{\ddagger}(\theta^{\ddagger}_{c}(\theta^{\star}))$ vs. $\theta^{\star}$. The "double-dagger curve before vertical translation" shows the vertical distance between the curves when the horizontal correspondence is applied to the data without computing $\hat{z}_0$. Equ. 9 averaged several measurements of this vertical distance to calculate $\hat{z}_0$, which resulted in the double-dagger curve matching the star curve vertically as well as horizontally.

7 Conclusion

The terrain matching problem has been described in the context of autonomous vehicle navigation over a $2\frac{1}{2}$-D unstructured terrain, such as the surface of Mars.

We have developed a new approach based on the max-min principle, even though exceptions are possible when using realistic curves that have noncorresponding endpoints. We use normal-to-intersect calculations in place of discrepancy measurements to simplify the computation. If we do encounter max-min exceptions in future experiments, a second derivative check will be inserted to insure that the normal-to-intersect measurements are local minima as opposed to local maxima.

We have described in detail the max-min calulations using normal-to-intersects, and how the translation is calculated from the max-min. The framework of the overall terrain matching algorithm was also described.

The method introduced here could be classified as "feature matching" as opposed to "iconic matching," in the terminology of [9]. It also has a fundamental distinction from the "cost function" approaches, for instance [4] and [1, sect. 5], which evaluate a cost function between the two sets of data in a grid of trial vantages, and update the vantage based on those samples of the cost function. Although the method introduced here may also require repeated iterations of vantage estimation when applied to noisy skylines, each iteration updates the vantage estimate based on only a single prior estimate, and not a grid of trial vantages.

Future publications will demonstrate the approach using two whole sets of skylines, instead of a single pair as described here. The skylines will be generated from a global map, using the algorithm described in [1, sect. 4].

References

[1] C. N. Shen and Lance Page. Fusion of gross satellite sensing and laser measurements by skyline map matching for autonomous unmanned vehicle navigation. In *Proc. SPIE Optical Engineering and Aerospace Sensing Symposium*, Orlando, Florida, April 1990.

[2] C. N. Shen and George Nagy. Autonomous navigation to provide long-distance surface traverses for mars rover sample return mission. In *Proc. Fourth IEEE International Symposium on Intelligent Control*, pages 362–367, Albany, NY, September 1989.

[3] C. N. Shen. Autonomous navigation for mobile robot vehicles over hilly terrain using rangefinding measurements. In *Proc. Robotic Intelligence and Productivity Conference*, Detroit, Michigan, November 1983.

[4] Donald Gennery. Visual terrain matching for mars rover. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, California, June 1989.

[5] Eric Krotkov. Mobile robot localization using a single image. In *Proc. IEEE Robotics and Automation Conference*, pages 978–983, 1989.

[6] C. Ming Wang. Location estimation and uncertainty analysis for mobile robots. In *Proc. IEEE Robotics and Automation Conference*, pages 1230–1235, 1988.

[7] Paul R. Klarer. Autonomous land navigation in a structured environment. *IEEE Aerospace and Electronic Systems Magazine*, 5(3):9–11, March 1990.

[8] William H. Press, Brian R. Flannery, Saul A. Taukolsky. and William T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1988.

[9] Hebert, Kanade, and Kewen. 3-D vision techniques for autonomous vehicles. Technical report, Carnegie Mellon University, August 1988. CMU-RI-TR-88-12.

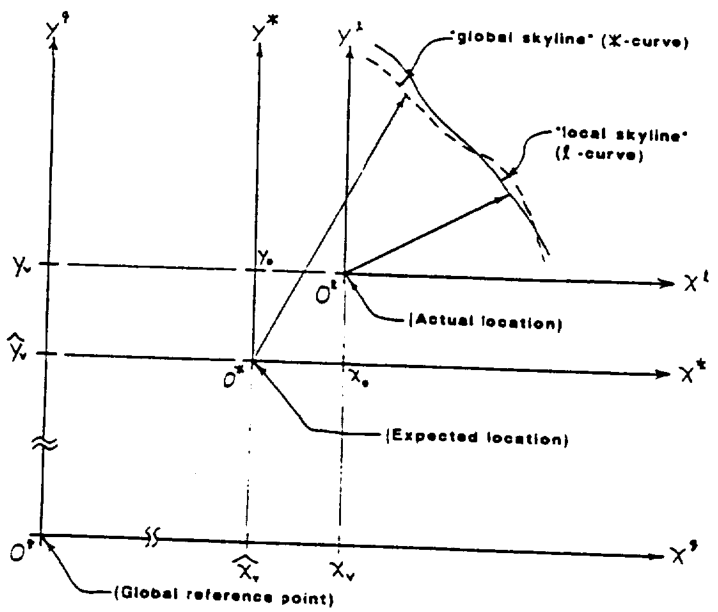


Figure 1: Relationships among the global (g), local (l), and star ($\star$) coordinate systems.

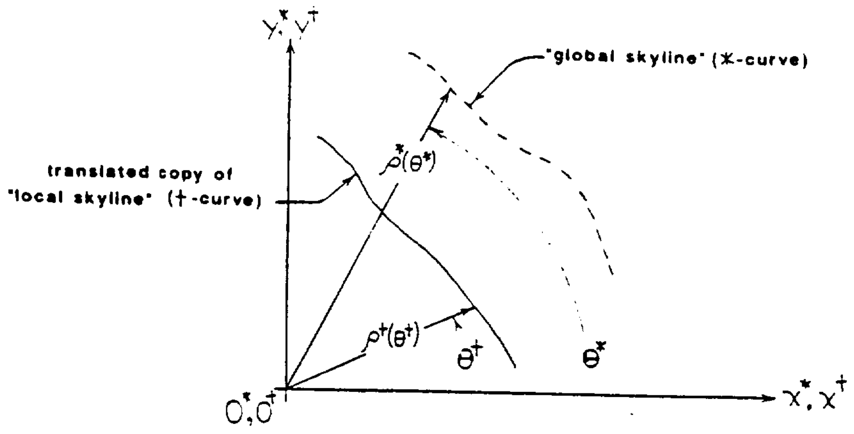

"global skyline" (✱-curve)
translated copy of "local skyline" (†-curve)

Figure 2: Illustration of the dagger (†) coordinate system.

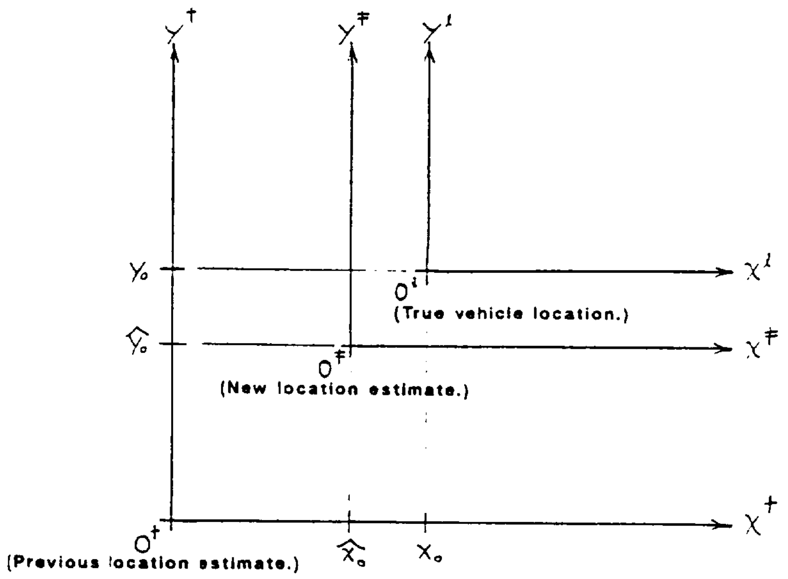


Figure 3: Three versions of the "local" system: the local (l), dagger (†), and double-dagger (‡) coordinate systems.

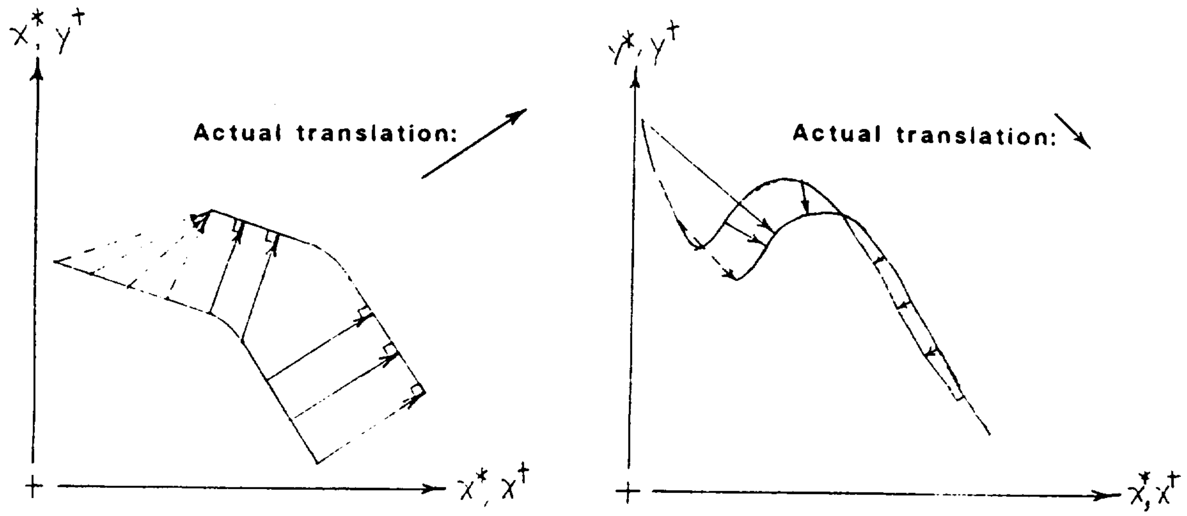


Figure 4: Max-min exceptions due to noncorresponding endpoints: on the left, discarding the discrepancies which terminate on endpoints is sufficient, but on the right it is not. (Dashed vectors represent discarded discrepancies.)

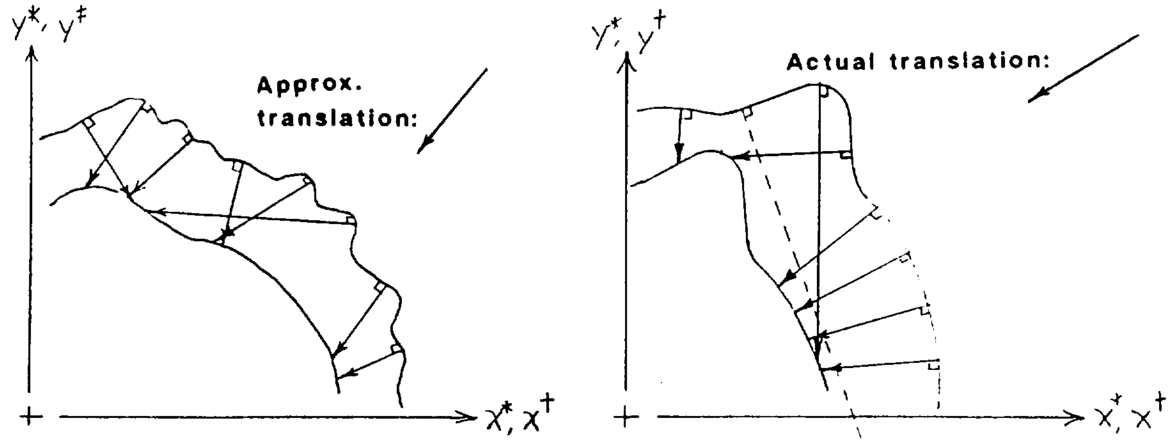


Figure 5: Max-min exceptions due to using normal-to-intercepts on noisy or sharply turning data.

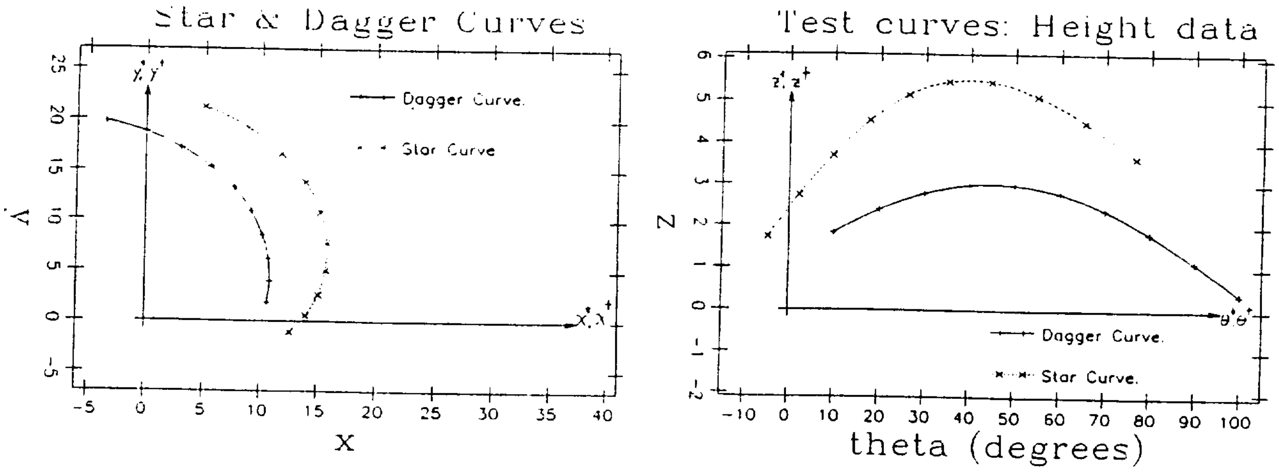
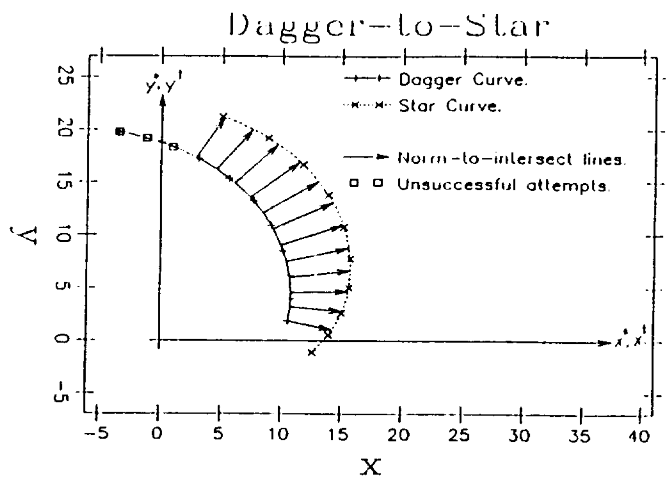


Figure 6: Star and dagger curves for test case.



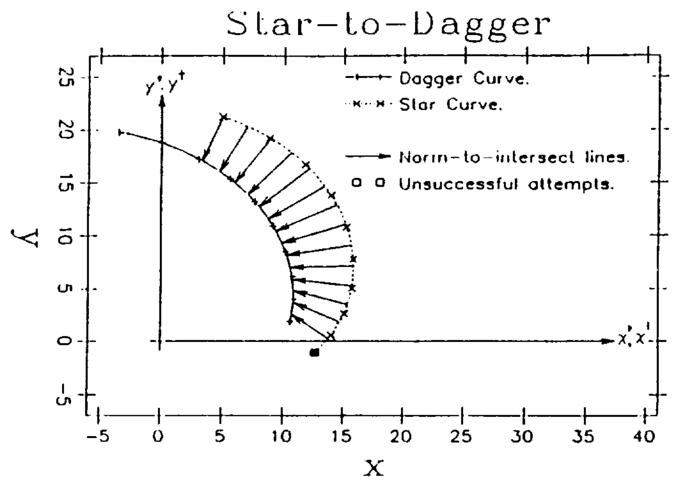
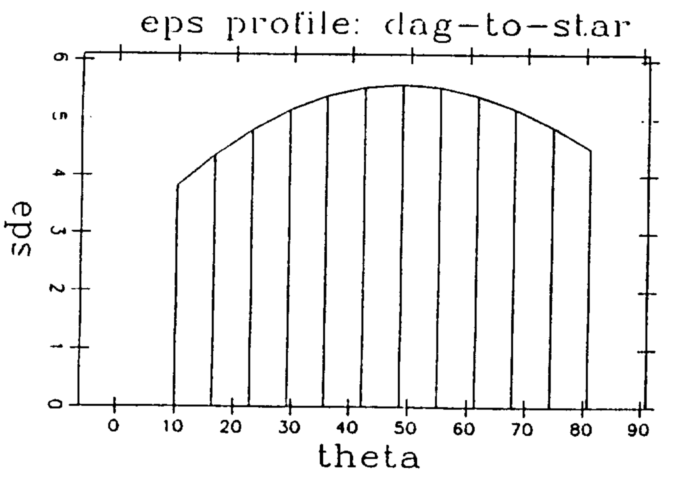


Figure 7: Normal-to-intersect vectors in both directions.



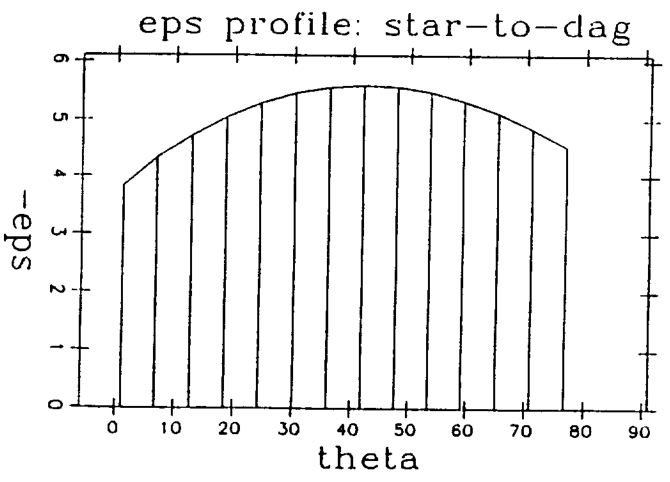


Figure 8: Epsilon profiles for experimental curves.

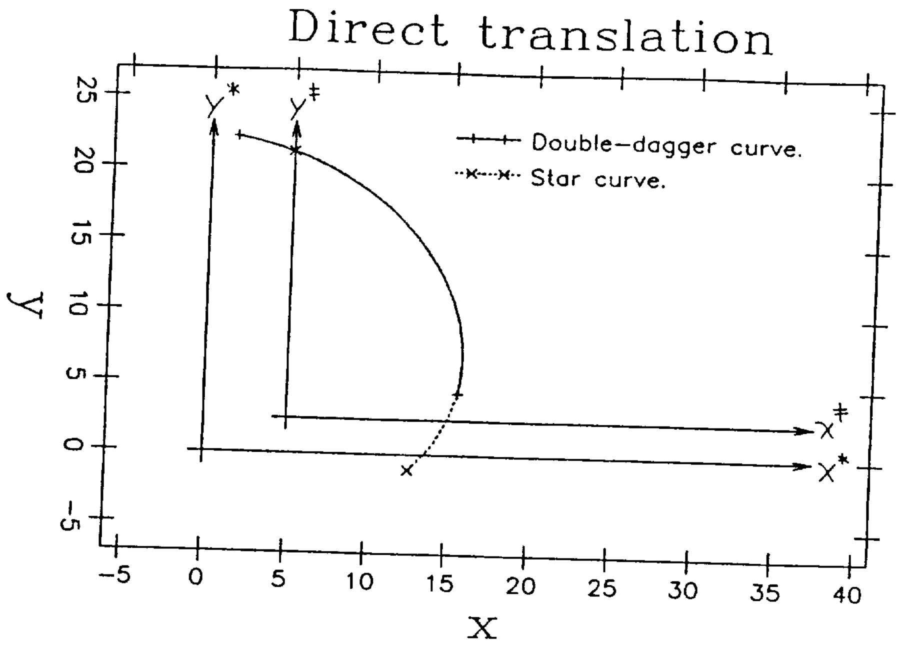


Figure 9: Curve match based on direct max-min.

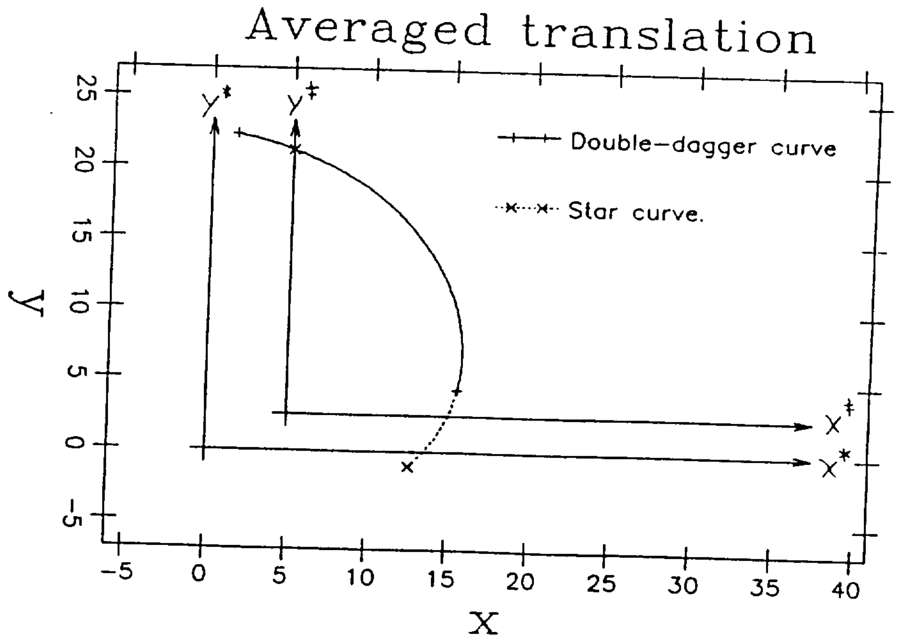


Figure 10: Curve match based on direction-to-intersect averaging.

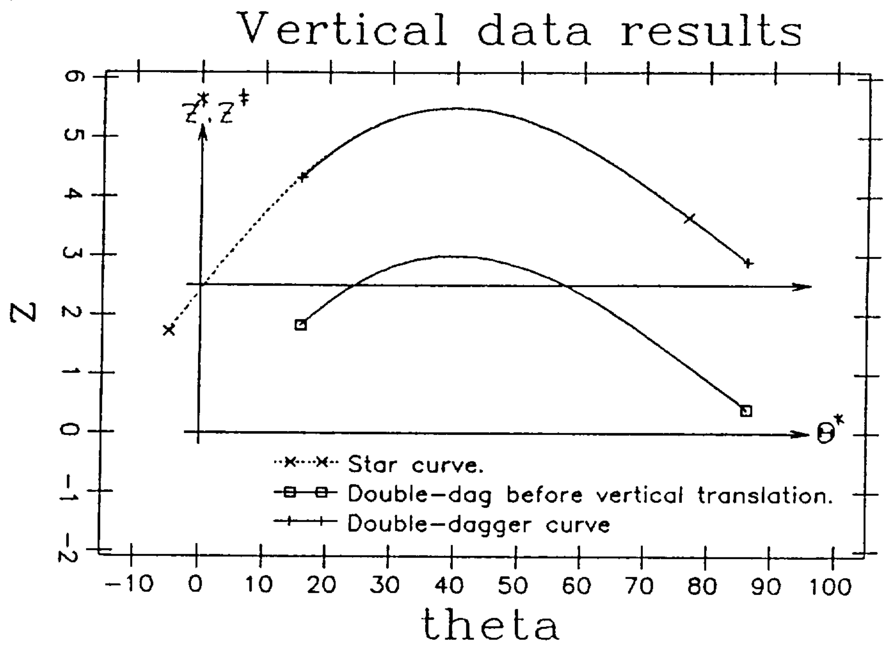


Figure 11: Matching of the vertical data, based on the correspondence developed from the horizontal data.